Alex Lindroos

Android-sovellus Kotlin-ohjelmointikielellä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

23.4.2018

Metropolia

Tekijä Otsikko Sivumäärä Aika	Alex Lindroos Android-sovellus Kotlin-ohjelmointikielellä 30 sivua + 5 liitettä 23.4.2018
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	tieto- ja viestintätekniikka
Ammatillinen pääaine	Mobile Solutions
Ohjaaja	yliopettaja Petri Vesikivi

Insinöörityön tarkoituksena oli tutustua sovelluksen kehittämiseen Androidille käyttäen Kotlin-ohjelmointikieltä ja verrata sitä kehittämiseen Java-ohjelmointikielellä. Työssä toteutettiin Reddit-uutissovellus Androidille käyttäen Kotlin-ohjelmointikieltä. Sovelluksella pystytään lukemaan suosituimpia julkaisuja ja kommentteja, tarkastelemaan omia tilattuja keskustelukanavia, etsimään keskustelukanavia ja katsomaan omia profiilitietoja. Sovellus kommunikoi Redditin rajapinnan kanssa, ja ohjelmointiympäristönä kehityksessä oli Android Studio.

Kotlin on Androidin uusi virallinen ohjelmointikieli, joka sisältää paljon uusia ja ohjelmointia helpottavia ominaisuuksia. Insinöörityössä pureuduttiin itse kieleen, ja muutamiin sen hyödyllisimpiin ominaisuuksiin. Kotlinia vertailtiin insinöörityössä myös Java-ohjelmointikieleen. Kotlinilla ohjelmoiminen on miellyttävää ja sen avulla sovelluksen koodin määrä on pienempi ja koodi on turvallisempaa kuin muilla Androidin virallisilla ohjelmointikielillä. Koska Kotlin on uusi ohjelmointikieli, on sen dokumentaatio vähäisempää verrattuna Javaan, ja tämä on syytä ottaa huomioon sillä kehittäessä.

Sovelluksen teossa perehdyttiin OAuth 2.0 -autorisointiprotokollaan ja sen käyttämiseen tunnistautumisessa sekä rajapinnan kutsujen tekoon ja MVP-mallin (Malli-näkymä-esittäjä) käyttämiseen arkkitehtuurissa. MVP-mallia käytettiin sovelluksessa, koska se on helposti laajennettavissa ja ylläpidettävissä. Sovelluksessa perehdyttiin myös sovelluksen ominaisuuksien kehittämiseen ja siihen, mikä on Reddit-uutispalvelu ja mitä siellä voi tehdä.

Kotlinin ominaisuuksia hyödyntäen insinöörityön sovelluksessa saatiin toteutettua helppokäyttöinen ja nopeasti toimiva Android-sovellus. MVP-mallin ja hyvien kirjastojen käyttäminen sovelluksessa mahdollisti järjestelmällisen ja laajennettavan kokonaisuuden.

Avainsanat	Kotlin, Reddit, Android, mobiilikehitys

Author	Alex Lindroos
Title	Android application made with Kotlin-programming language.
Number of Pages	30 pages + 5 appendices
Date	23 April 2018
Degree	Bachelor of Engineering
Degree Programme	Information and Communication Technology
Professional Major	Mobile Solutions
Instructor	Petri Vesikivi, Principal Lecturer

The purpose of this thesis was to develop a Reddit client for the Android devices using Kotlin as a development language. Application can browse and read the most popular posts and comments, search subreddits, see your own subreddits and profile information. It is made by using Android Studio.

Kotlin is a new statically typed programming language made by JetBrains. It was announced as an official programming language for Android in 2017.

Thesis also wraps up some of the best Kotlin features and comparison with Java language. Thesis goes through Android and its architecture and architecture patterns MVC (Model-view-controller) and MVP (Model-view-presenter). Rest of the thesis covers the development and design process of the application. It will also provide information about OAuth 2.0 authorization and how is it used in the application and networking with the Reddit API (Application Programming Interface).

Thesis provides detailed explanation for the features and how they are implemented in the application. It will also explain what is Reddit and what can you do there.

Using Kotlin for the development of the application, it benefits the developer to make a fast and easy to use application. Also using the right libraries and MVP-pattern makes the whole application expandable and responsive.

Keywords	Kotlin, Reddit, Android, mobile development

Sisällys

Lyhenteet

API	Application Programming Interface on ohjelmointirajapinta, joka tässä projektissa on Redditin.
MVC	Model-View-Controller on malli-näkymä-ohjain-arkkitehtuurimalli.
MVP	Model-View-Presenter on malli-näkymä-esittäjä-arkkitehtuurimalli.
APK	Android application package on Android-sovellusten tiedostotyyppi.
JVM	Java Virtual Machine eli Java-virtuaalikone suorittaa Javan bittikoodia.
DEX	Dalvik Executable on Dalvik-virtuaalikoneen bittikoodin tiedostotyyppi.
GC	Garbage collector on automaattinen muistinhallintamekanismi.
HTTP	Hypertext Transfer Protocol on hypertekstin siirtoprotokolla.
CRUD	Create, Read, Update, Delete tarkoittaa luo, lue, päivitä ja kirjoita.
URI	Uniform Resource Identifier, kertoo tietyn tiedon sijainnin.
SDK	Software Development Kit tarkoittaa ohjelmiston kehittämiseen luotuja työkaluja.
REST	Representational State Transfer on arkkitehtuurimalli ohjelmointirajapintojen toteutukseen.

1 Johdanto

Insinöörityön tarkoituksena on tutkia Android-sovelluksen tekoa Kotlin-ohjelmointikielellä, ja tavoitteena on tehdä sen avulla Reddit-uutispalvelun lukuun tarkoitettu sovellus Androidille. Ohjelmointiympäristönä kehityksessä on Android Studio. Työssä esitellään Kotlinin vahvuuksia ja se, miten se eroaa suositusta Java-ohjelmointikielestä. Sovelluksen teossa perehdytään OAuth 2.0 -autorisointiprotokollan käyttämiseen tunnistautumisessa, API-kutsujen tekoon, natiivien Android-komponenttien käyttöön, epäsynkronoitujen kutsujen hallitsemiseen RXJavalla ja MVP-mallin käyttämiseen arkkitehtuurissa.

Valitsin tämän aiheen, koska minulla on kokemusta natiivin Androidin kehittämisestä ja viime aikoina olen käyttänyt paljon Kotlinia. Kotlin on vakuuttanut minut ja monet muut siitä, miten paljon helpompi ja monipuolisempi ohjelmointikieli se on verrattuna Javaan. Reddit-uutispalvelun valitsin sovellukseen, koska sillä on laaja API-kanta ja hyvät dokumentaatiot.

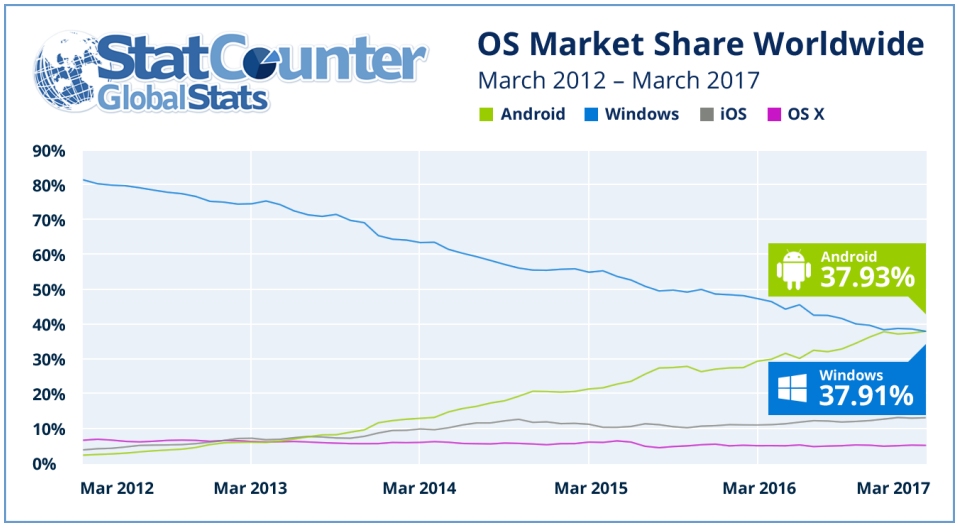
Sovellus on tarkoitettu Reddit-uutispalvelun lukemiseen. Sovellukseen kirjaudutaan omilla Reddit-käyttäjätunnuksilla, ja sen jälkeen käyttäjä pystyy lukemaan suosikkikeskustelukanaviaan. Käyttäjä pystyy myös etsimään keskustelukanavia, katsomaan parhaat kommentit ja tarkastelemaan omia käyttäjätietoja.

2 Android-mobiilikäyttöjärjestelmä

2.1 Androidin historia

Android on Googlen kehittämä mobiilikäyttöjärjestelmä, jonka pohjana on Linux-järjestelmä, ja se on rakennettu nimenomaan puhelimille ja tableteille. Se sai alkunsa vuonna 2003, kun Nick Sears, Rich Miner, Chris White ja Andy Rubin perustivat yhdessä Android Inc. -nimisen yrityksen. Kaksi vuotta tämän jälkeen Google osti yrityksen. [1.]

Vuonna 2008 julkaistiin ensimmäinen Android-mobiilikäyttöjärjestelmän sisältävä puhelin. Se oli nimeltään HTC Dream tai Yhdysvalloissa ja muutamissa Euroopan maissa T-Mobile G1. [2.] Puhelimessa oli laadukas kamera, liu`utettava näppäimistö ja siihen aikaan hyvät osat. Puhelin sisälsi ensimmäisen Android-ohjelmistoversion 1.0, joka mullisti maailman mobiilimarkkinoita, sillä se sisälsi ominaisuuksia, joita nykyään on joka puhelimessa, esimerkiksi pienoisohjelmat ja ilmoitukset [2]. Julkaisun jälkeen Android on tehnyt useita käyttöjärjestelmän versioita, joista uusimpana on Oreo 8.1, joka julkaistiin elokuussa 2017.



Kuva 1. Käyttöjärjestelmien markkinaosuus internetin käytössä maaliskuu 2012 – maaliskuu 2017 [3].

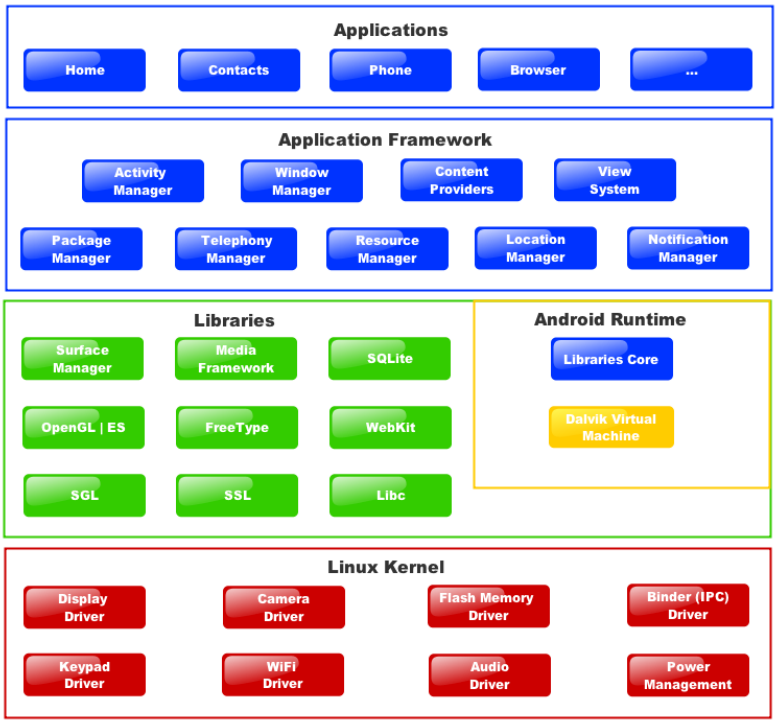
Kuvassa 1 on esitetty käyttöjärjestelmien markkinaosuus internetin käytössä aikavälillä maaliskuu 2012 ja maaliskuu 2017. Tarkoituksena oli verrata eri käyttöjärjestelmien suosiota lopullisesta internetin käytöstä matkapuhelimien, pöytätietokoneiden, kannettavien

ja tablettien kesken. Vertailussa oli mukana neljä eri käyttöjärjestelmää. Android, Windows, iOS ja OS X. Maaliskuussa 2017 Android oli ohittanut Windowsin kahdella prosentilla. Tämä kertoo sen, kuinka Android-käyttöjärjestelmien käyttö on jatkanut noususuhdannettaan ja ne ovat vieläkin mobiililaitteiden suosituin käyttöjärjestelmä. [3.] Android on avoimen lähdekoodin käyttöjärjestelmä, joten kuka vain voi tehdä siihen perustuvan käyttöjärjestelmän. Tämä on yksi syistä, joiden takia Android on käytössä niin monessa mobiililaitteessa.

2.2 Android-arkkitehtuuri

Android-käyttöjärjestelmä on ohjelmistopino (kuva 2), joka on jaettu viiteen tasoon:

- sovellukset
- sovelluskehys
- kirjastot
- Linux-ydin
- Android Runtime.



Kuva 2. Androidin ohjelmistopino [32].

Kuvassa näkyy Androidin ohjelmistopino ja mitä eri tasot pitävät sisällään. Jokainen taso on yhteyksissä toisiinsa, ja tämä kokonaisuus mahdollistaa käyttöjärjestelmän toimivuuden. Seuraavaksi tasoja käsitellään tarkemmin.

Linux-ydin

Linux-ydin on Androidin ohjelmistopinon alin kerros, ja se vastaa käyttöjärjestelmän muistista, säikeistä, ohjelmistoajureista ja turvallisuudesta. Tätä tasoa pidetään yhtenä tärkeimmistä ohjelmistopinon tasoista [4]. Jokaisella käyttöjärjestelmällä on ydin, jonka kanssa sovellukset ja laitteisto keskustelevat. Kun sovellus tekee minkä vain toiminnon, se lähettää pyynnön ytimelle ja toimii vastauksen mukaisesti [5].

Kirjastot

Androidin natiivikirjastot sijaitsevat Linux-ydintason yläpuolella. Natiivikirjastot vastaavat erilaisten datojen käsittelystä [4]. Kirjastot on kirjoitettu C- tai C++-kielellä. Esimerkiksi datan tallennuksesta vastaa SQLite-kirjasto ja muina esimerkkeinä ovat Webkit-kirjasto,

joka on avoimen lähdekoodin selain, ja mediasovelluskehys, jonka tehtävä on käsitellä kaikkea ääni- ja videodataa [6].

Android Runtime

Android Runtime on ajonaikainen taso Androidin ohjelmistopinossa, ja se on samalla tasolla kirjastojen kanssa. Se pitää sisällään kaksi komponenttia:

- **Ydinkirjastot** ovat Dalvik-virtuaalikoneelle ja muille sovelluksille merkittäviä. Ne muodostavat sovelluskehyksen perusteet. Sovellukset käyttävät niitä elintärkeissä toiminnoissaan, esim. verkon käytössä, tiedostojen hallinnassa ja grafiikassa. [7.]

- **Dalvik-virtuaalikone** vastaa sovellusten suorittamisesta rajoitetussa tilassa [7]. Se muuntaa sovellusten Java-bittikoodia dex-bittikoodiksi, jota virtuaalikone käyttää. Nämä dex-tiedostot kootaan myöhemmin yhdeksi apk-tiedostoksi, joka on Androidin sovellusten tiedostotyyppi. Dalvik-virtuaalikone on suunniteltu myös niin, että se pystyy ajamaan useita virtuaalikoneita samanaikaisesti. Android 4.4 KitKat -ohjelmistoversiossa Google esitteli ensimmäistä kertaa ART-ympäristön, jonka tarkoitus on syrjäyttää vanha Dalvik-virtuaalikone. ART tarjoaa parannuksia nopeuteen sovellusten ajoissa ja latauksissa. [8.] ART:n ominaisuuksiin kuuluu myös parannettu GC (Garbage Collector) ja uudistetut kehittäjien työkalut.

Sovelluskehys

Androidin sovelluskehys on työkaluja, jotka Androidin kehittäjät ovat tehneet helpottaakseen sovellusten kehittämistä ja kommunikointia Android-laitteiden kanssa [9, s. 28]. Tärkeimmät sovelluskehyksen työkalut ovat seuraavat:

- **Aktiviteetinhallinta** hallitsee aktiviteettien elämänkaarta ja sitä, miten ne pinotaan taustalla.

- **Paketinhallinta** tarjoaa tietoa laitteelle asennetuista paketeista ja niiden luvista.

- **Ikkunoidenhallinta** vastaa ikkunoista ja niiden järjestyksestä sekä siitä, mikä ikkuna näkyy milläkin hetkellä näytöllä.

- **Palveluntarjoajat** hallitsevat sovelluksen dataa ja pystyvät liikuttelemaan sitä eri sovellusten välillä.

- **Näkymänhallinta** vastaa sovelluksen näkymistä ja näkymäryhmistä. Näkymä on objekti, joka piirtää jotakin näytölle ja mahdollistaa käyttäjän vuorovaikutuksen. Kaikki sovellukset rakentuvat näkymistä ja niiden ryhmistä.

- **Puhelinpalveluidenhallinta** tarjoaa saatavilla olevien puhelinpalveluiden tilaa ja muita tietoja.

- **Resurssienhallinta** vastaa kaikista sovelluksen resursseista, esimerkiksi kielitiedostoista, väreistä ja ulkoasuista.

- **Sijainninhallinta** vastaa nimensä mukaisesti laitteen sijaintitiedoista.

- **Ilmoitustenhallinta** hallitsee sovellusten ilmoituksia ja kertoo niillä taustalla ajettavien sovellusten tapahtumista. [6.]

Sovellukset

Sovelluskerros on Androidin arkkitehtuurin ylin kerros, ja se tarjoaa joukon ydinohjelmia, niin kuin esimerkiksi sähköpostin, tekstiviestit, internetselaimen ja kontaktit. Androidin omat sovellukset pystyvät käyttämään sovelluskerroksen sovelluksia toiminnoissaan, ja kehittäjät voivat hyödyntää niitä omissaan. Tämä helpottaa sitä, että kehittäjien ei tarvitse tehdä itse ominaisuuksia, joilla pääsee toimintoihin käsiksi.

2.3 Sovelluksen rakenne

Androidille tehtäviä sovelluksia pystytään kirjoittamaan Androidin virallisilla kielillä, Kotlinilla, Javalla ja C-kielillä [10]. Ohjelmointiympäristönä suosituin on Android Studio, joka on Googlen tekemä virallinen ohjelmointiympäristö Androidille ja jonka käyttäminen on helppoa. Jokainen Android-sovellus ajetaan omassa Linux-prosessissaan ja käyttää uniikkia Linux-tunnusta, jonka antaa Androidin järjestelmä. Linux-tunnuksella järjestelmä

pystyy tunnistamaan sovelluksen ja tarjoamaan sille kuuluvat oikeudet ja tiedot. Sovellukset toimivat myös omissa virtuaalikoneissaan ja ovat näin ollen eristyksissä muista sovelluksista. [10.]

Sovellukset rakentuvat erilaisista Androidin komponenteista, jotka ovat Android-sovelluksien perusta. Komponentteja on neljä erilaista, ja jokaisella komponentilla on myös oma elinkaarensa, josta Androidin järjestelmä vastaa. [10.]

Aktiviteetit

Aktiviteetit ovat sisääntuloja käyttäjän ja sovelluksen interaktioiden välillä. Aktiviteetti on yksi käyttöliittymällä varustettu näkymä. Sovellus koostuu yleensä useista aktiviteeteista. Aktiviteetit vastaavat siitä, mitä käyttäjä näkee näytöllä, ja pystyvät hallitsemaan elinkaarien eri metodien avulla, mitä tehdään esimerkiksi, kun käyttäjä avaa sovelluksen ja sulkee sen. Aktiviteetin saa käyttöön luokalle lisäämällä Androidin oman Activity-nimisen luokan ja tämän myötä lisäämällä halutut elinkaarimetodit. [10.]

Aktiviteetilla on erilaisia elinkaari-metodeja (kuva 3), joilla pystyy kontrolloimaan, mitä tapahtuu erilaisissa tilanteissa, kun aktiviteetti on avattu, suljettu tai pysäytetty:

- **OnCreate()**-metodi kutsutaan heti, kun aktiviteetti luodaan. Sen sisällä luodaan aktiviteetin näkymät, ja siellä yleensä myös suoritetaan tietojen alustus. Se on myös ainoa pakollinen lisättävä elinkaari-metodi, sillä sen avulla näytetään aktiviteetin näkymä. [12.]

- **OnStart()**-metodin vastuulla on aktiviteetin näkymän näyttäminen käyttäjälle. Sen sisällä kannattaa suorittaa kaikki käyttöliittymän alustukset. [12.]

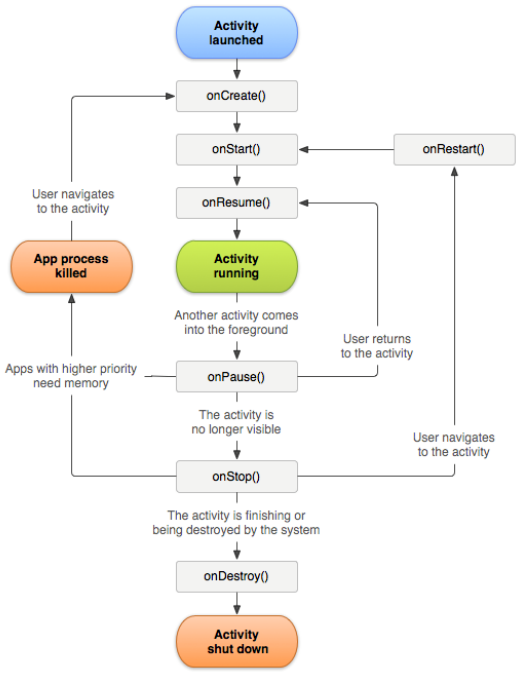
- **OnResume()**-metodia kutsutaan heti OnStart()-metodin jälkeen, ja tämän jälkeen käyttäjä voi aloittaa aktiviteetin käyttämisen. [12.]

- **OnPause()**-metodia kutsutaan heti, kun käyttäjä poistuu aktiviteetista tai jokin muu sovellus keskeyttää käyttämisen. OnPause-metodia käytetään yleensä muistin vapauttamiseen. Jos aktiviteetti käyttää esimerkiksi sensoreita tai sillä on taustapalveluita käytössä, OnPause-metodissa on hyvä lopettaa ne. Näin säästetään muistia, akkua ja muita resursseja. Androidin järjestelmä pitää aktiviteetin

muistissa, joten palattaessa aktiviteettiin sillä on tallessa näkymät ja tiedot, jotka näkymään jäivät. [12.]

- **OnStop()**-metodia kutsutaan, kun näkymä ei näy käyttäjälle, esimerkiksi kun käyttäjä on mennyt toiseen sovellukseen. Tässä metodissa voi sulkea toimintoja, joita ei tarvita, ja vapauttaa sitä myöten resursseja. [12.]

- **OnDestroy()**-metodi tuhoaa koko aktiviteetin ja sulkee kaikki sen toiminnot [12].



Kuva 3. Aktiviteetin elinkaari [11].

Palvelut

Yksi Android-sovelluksen osa-alueista on palvelut. Palvelut ovat sovelluksen taustalla ajettavia komponentteja ilman käyttöliittymää. Palveluilla on oma elinkaarensa, ja ne

operoivat isäntäprosessin pääsäikeessä. Palvelu on sidoksissa sen omaan säikeeseen, jossa se toimii. Palvelun alkaessa se luo itselleen säikeen, ja kun säie on valmis, se ilmoittaa palvelulle, että se on valmis. [15.]

Palveluita on kolmea erilaista tyyppiä: etualalla toimivat palvelut, taustalla toimivat palvelut ja sidotut palvelut. Etualalla toimivat palvelut ovat näkyvissä käyttäjälle. Esimerkiksi sovellus, joka näyttää videoita, voi käynnistää etualalla toimivan palvelun, kun käyttäjä katsoo videoita. Etualalla toimivien palveluiden on oltava näkyvissä puhelimen näytön yläpalkissa. Taustalla toimivat palvelut toimivat nimensä mukaisesti taustalla, esimerkiksi palvelu, joka lähettää palvelimelle tietoja käyttäjän sijainnista sovelluksen ollessa käynnissä. Sidotut palvelut on sidottu johonkin sovelluksen komponenttiin, ja niillä on käyttöliittymä käyttäjän ja palvelimen välille. Tämä mahdollistaa käyttäjän pääsyn palvelun tietoihin ja tuloksiin. [16.]

Broadcast Receiver

Broadcast Receiver on komponentti, joka mahdollistaa sovelluksen ulkopuolisten ilmoitusten vastaanottamisen. Sen käyttötarkoitus voi esimerkiksi olla sellainen, että sovellus haluaa tietää, milloin puhelimen akku on loppumassa. Broadcast Receiver kuuntelee järjestelmän tarjoamaa tietoa akun varauksesta ja vastaanottaa siltä ilmoituksen, kun akun virta on vähissä. Tämä laukaisee tapahtuman sovelluksessa, ja tämä mahdollistaa jonkin tietyn asian tekemisen tämän jälkeen. [10.]

Palveluntarjoajat

Palveluntarjoajat vastaavat sovelluksen tietojen ja tiedostojen tarjoamisesta muille sovelluksille. Ne toimivat neljällä perustoiminnolla, jotka ovat luo, lue, päivitä ja poista. Näitä toimintoja kutsutaan nimellä CRUD (Create, Read, Update, Delete). Palveluntarjoajat käyttävät tiedostolle tehtyä URI:a sen löytämiseen ja lähettämiseen (esimerkkikoodi 1). Se koostuu tarjoajan nimestä ja tyypistä, jotka osoittavat tiedoston polkuun. [10.]

```
content://media/external/images/media/010101
```

Esimerkkikoodi 1. Palveluntarjoajan URI-esimerkki.

Palveluntarjoajat mahdollistavat myös toisen sovelluksen komponenttien ja ominaisuuksien käyttämisen omassa sovelluksessaan. Ne pystyvät pyytämään, vaikka äänennauhoitussovelluksen äänitekomponentin käyttämistä omassa sovelluksessaan. Sovelluksen on tehtävä tällöin pyyntö Androidin järjestelmälle, joka avaa komponentin sovellukselle. [10.]

2.4 Arkkitehtuurimallit

Arkkitehtuurimallit ovat tapa rakentaa sovelluksia niin, että ne ovat skaalautuvia, toimivat sujuvammin ja niitä on helpompi huoltaa tai muokata tulevaisuudessa. Sovelluksen koon kasvaessa arkkitehtuurimallin implementoinnin tärkeys kasvaa. Arkkitehtuurimalleja on monenlaisia, ja kehittäjien on hyvä vertailla, mikä malli sopii sovellukseen. Seuraavaksi käydään läpi kaksi tunnettua arkkitehtuurimallia erityisesti mobiiliohjelmoinnissa.

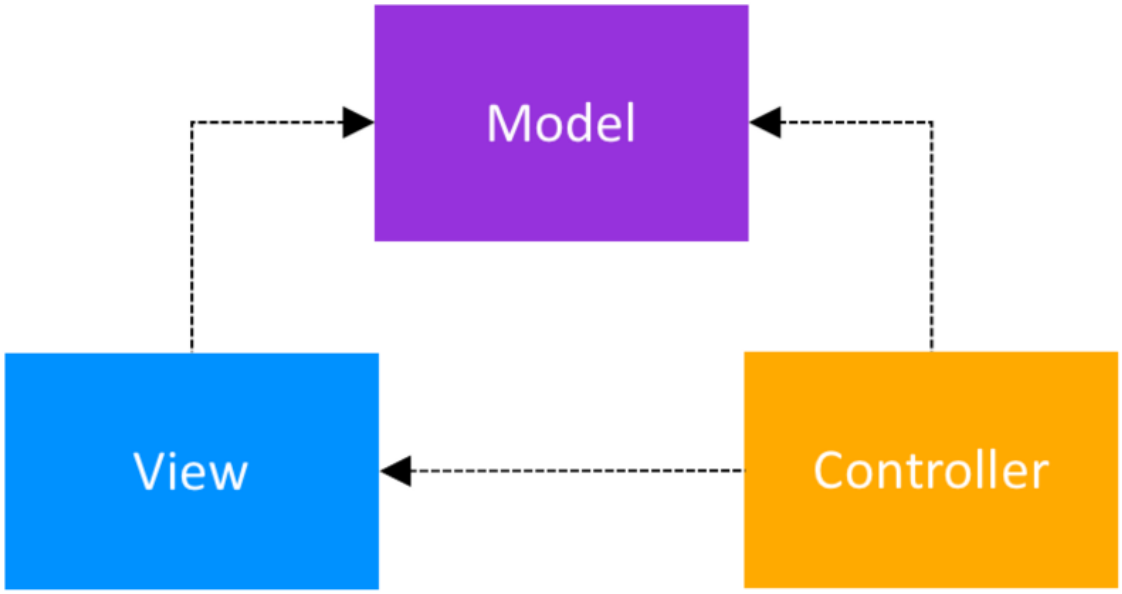
MVC-arkkitehtuuri

MVC-malli (malli-näkymä-ohjain) (kuva 4) on arkkitehtuurimalli, jota käytetään erityisesti sovelluksissa, joissa on käyttäjien interaktioita [18].

Näkymä on sovelluksen käyttöliittymä. Käyttäjän interaktiot siirtyvät ohjaimelle, joka käsittelee ne. [19.]

Ohjain hallinnoi sovellusta. Se vastaanottaa käyttäjäinteraktion näkymältä ja tekee tarvittavat toimenpiteet ja päivittää mallin. [19.]

Malli edustaa sovelluksen dataa, ja se vastaanottaa käskyjä ohjaimelta. Käskyn saadessaan se päivittää itseään sen mukaan ja ilmoittaa näkymälle muutoksen. [19.]



Kuva 4. MVC-malli [17].

Androidissa MVC:n näkyminä toimivat aktiviteetit ja fragmentit. Ohjaimena taas toimii erillinen luokka, jolla on referenssi näkymään eli tässä tapauksessa aktiviteettiin tai fragmenttiin. Helpoimmin tämä referenssi saadaan, kun tehdään näkymille pohjaluokka, jonka kaikki näkymät perivät. Tähän pohjaluokkaan on ohjaimella referenssi. [19.]

Kun logiikan hajauttaa arkkitehtuurimalleihin, on kehittäjän paljon helpompi paikantaa vika ja lisätä uusia ominaisuuksia. Tämän arkkitehtuurimallin hyvinä puolina on myös koodin testattavuus. Mallilla ja näkymällä ei ole referenssejä mihinkään Androidin omiin luokkiin, joten tämä mahdollistaa niiden yksikkötestaamisen [19]. MVC:n ongelmana on useasti ohjainluokka, joka kasvaa liian suureksi ja skaalautuvuus on heikkoa.

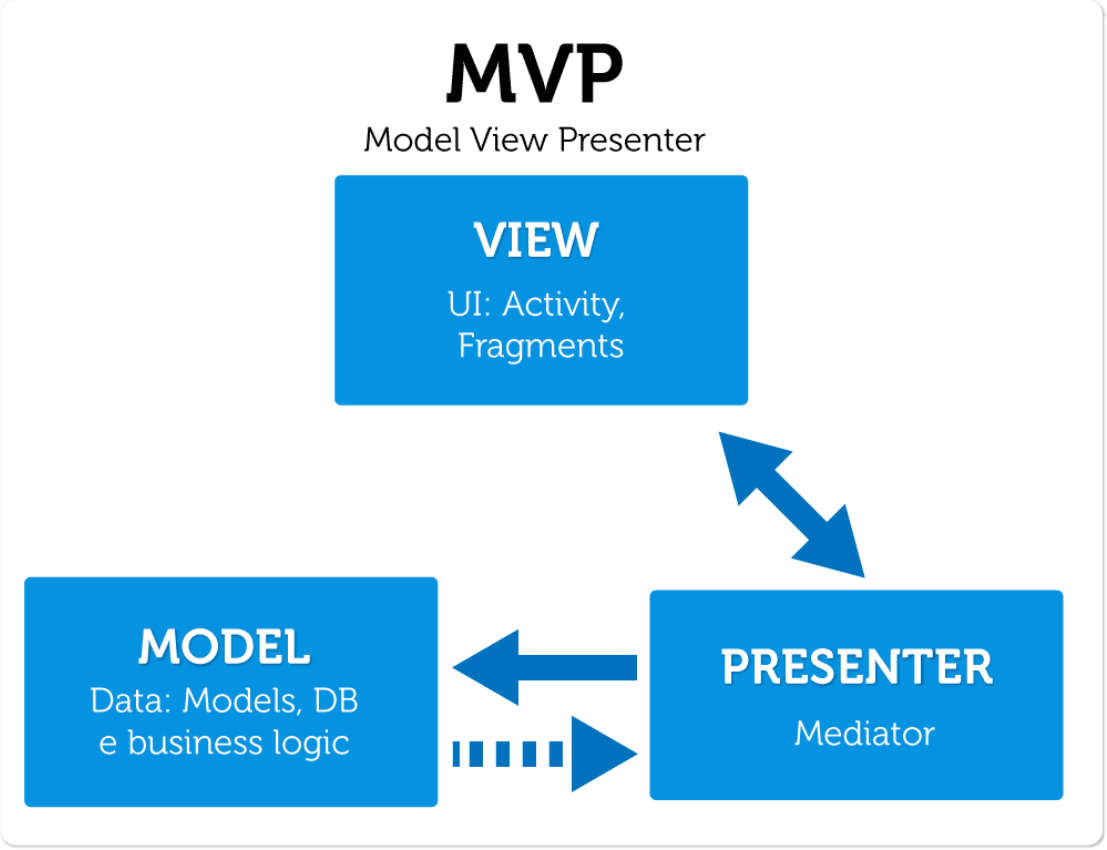
MVP-arkkitehtuuri

MVP-malli (malli-näkymä-esittäjä) (kuva 5) on toinen tunnettu arkkitehtuurimalli sovelluskehityksessä. Insinöörityön sovellus käyttää tätä arkkitehtuurimallia.

Malli on tässäkin arkkitehtuurimallissa sovelluksen datakerros. Malli kommunikoi mahdollisten tietokantojen ja API-kerrosten kanssa. Se kommunikoi vain esittäjän kanssa. [20.]

Näkymä on myös se, joka vastaa käyttäjien interaktioista ja esittäjän kanssa kommunikoinnista [20].

Esittäjä hoitaa kaiken logiikan. Se hallitsee näkymää ja datan hakemista mallista. [20.]



Kuva 5. MVP-malli [21].

Kun MVP-mallia käytetään Androidissa, on hyvä tehdä näkymästä passiivinen, niin että se vain ottaa esittäjän dataa vastaan ja vie sen aktiviteettiin tai fragmenttiin. Esittäjää tehdessä ei pidä laittaa sitä perimään eri kirjastoja, vaan pitää se puhtaana. Jos esittäjän pitää tehdä esimerkiksi API-kutsuja, on hyvä tehdä sille oma luokkansa ja antaa se esittäjälle parametrina. MVP-mallin käytössä pitää muistaa käyttää järkevää funktioiden nimeämistä. Tämä helpottaa koodin lukemista ja muokkaamista. Hyvänä muistisääntönä on miettiä, mitä funktio tekee, ja nimetä se sen mukaan. Esittäjä tarvitsee aina näkymän, joten on myös järkevää luoda esittäjälle pohjaluokka, joka tekee referenssin näkymään, kun aktiviteetti luodaan, ja poistaa referenssin, kun aktiviteetti tuhotaan. [26.]

MVP-mallilla on etuna MVC-malliin nähden se, että siinä ei ole kuin yksi luokka, joka hallitsee näkymän, ja se on tässä tapauksessa esittäjä. MVC-mallissa taas voi olla monta luokkaa, jotka voivat hallita näkymää. Tämä mahdollistaa vain yhden luokan yksikkötestauksen.

3 Kotlin-ohjelmointikieli

Kotlin on staattisesti tyypitetty ohjelmointikieli, jonka on kehittänyt JetBrains-niminen yritys. JetBrains tunnetaan parhaiten IntelliJ IDEA -ohjelmointiympäristön kehittämisestä [13]. Java-ohjelmointikielitaustan omaaville kehittäjille Kotlinin oppiminen on helppoa, sillä siinä on paljon yhtenäisyyksiä Javan kanssa. Kotlinista tuli Androidin virallinen ohjelmointikieli vuonna 2017, ja näin ollen Google aloitti sen täysimittaisen tukemisen uusissa Androidin ominaisuuksissa [13]. Se tuottaa Java-yhteensopivaa bittikoodia, eli sellaista, mikä toimii JVM:ssä (Java Virtual Machine).

Kotlinilla on useita vahvuuksia Java 6:een verrattuna, ja niitä käsitellään tässä projektissa. Kotlinilla pystyy myös kirjoittamaan ytimekkäämmin asioita, jotka vaatisivat Javalla paljon enemmän koodirivejä. Esimerkkikoodeissa 2 ja 3 näkyy User-niminen luokka Javalla ja Kotlinilla tehtynä.

```
public class User {

      private String firstname;
      private String lastname;
      private String userId;

public String getFirstname() {
      return firstname;
}
public void setFirstname(String firstname){
      this.firstname = firstname;
}
public String getLastname() {
      return lastname;
}
public void setLastname(String lastname){
      this.lastname = lastname;
}
public String getUserId() {
      return userId;
}
public void setUserId(String userId){
      this. userId = userId;
}
}
```

Esimerkkikoodi 2. User-luokan teko Javalla.

```
data class User (var firstname: String, var lastname: String, var userId:
String)
```

Esimerkkikoodi 3. User-luokan teko Kotlinilla.

Kotlin luo automaattisesti muuttujien hakuun ja asettamiseen tarvittavat metodit ja lisäksi antaa valmiita hyödyllisiä metodeita kehittäjän käyttöön. Javalla saman luokan kirjoittaminen vaatii paljon vakiokoodia samoille metodeille. [13.]

Koska Kotlin on staattisesti tyypitetty kieli, on jokaisella muuttujalla oltava määritelty tyyppi. Kotlin osaa tunnistaa joitakin tyyppejä automaattisesti. Jokainen muuttuja ei voi olla lähtökohtaisesti null-arvoinen eli arvoltaan tyhjä, vaan muuttujat, jotka voivat olla null, pitää merkitä vaihtoehtoiseksi lisäämällä kysymysmerkki tyypin perään ja antamalla arvoksi null.

Null-arvojen kanssa työskentely on Javalla aikaa vievää. Javalla pitää tarkistaa, onko jokin arvo null, ennen kuin sitä voi käyttää ilman virheilmoituksia ajonaikana, jos arvo onkin tyhjä. Kotlinissa pystyy antamaan muuttujan tyypille operaattorin, joka kertoo, että tämä muuttuja voi olla null ilman, että sovellus kaatuu ja antaa virheilmoituksen [13]. Kotlin pitää sisällään muitakin tapoja toimia null-arvojen kanssa. Turvallinen kutsuminen tehdään käyttämällä ”?.”-merkkiä. Tämän merkin jälkeen koodi ei etene, jos arvo ennen merkkiä on null. Esimerkkikoodissa 4 on nähtävissä, että jos car-muuttuja on null, koodia ei suoriteta, mutta jos car-muuttuja ei ole null, colorIs()-funktio suoritetaan. Tämä on turvallinen tapa käsitellä muuttujia, jotka voivat olla null-arvoisia, ja näin vältytään sovelluksen kaatumisilta.

```
car?.colorIs()
```

Esimerkkikoodi 4. Turvallinen tapa käsitellä koodia Kotlinissa.

Kotlin mahdollistaa myös uusien laajennoksien luomisen mihin vain luokkaan (esimerkkikoodi 5). Luokan ei tarvitse olla kehittäjän itsensä tekemä vaan se voi olla Androidin tai minkä vain kolmannen osapuolen luokka. Laajennokset ovat vain staattisia funktioita, jotka ottavat parametriksi sen luokan instanssin, jolle laajennos tehdään. Tämä mahdollistaa pääsyn luokan tietoihin funktion sisällä. Luokka ottaa kyseisen funktion parametrina ja sitä kutsuttaessa suorittaa sen. Laajennoksien luonti vähentää koodin määrää ja helpottaa ohjelmointia. [14.]

```
fun Fragment.toast(message: CharSequence, duration: Int = Toast.LENGTH_LONG) {
   Toast.makeText(this, message, duration).show()
}

fragment.toast("Hello")
```

Esimerkkikoodi 5. Laajennoksen luonti ja kutsuminen Kotlinilla.

Yksi koko ajan kehityksen alaisena olevista Kotlinin ominaisuuksista on Coroutines. Se on vaihtoehtoinen tapa käyttää säikeitä Android-kehityksessä. Android-sovellus käyttää yhtä pääsäiettä oletuksena, ja jos sovelluksessa suoritetaan operaatioita niin, että pääsäie estetään. Tämän seurauksena käyttöliittymä pysähtyy siihen asti, kunnes operaatio on suoritettu. Yleensä tähän on käytetty ratkaisuksi tehdä taustalla toimivia säikeitä, jotka eivät estä pääsäikeen toimintaa. Ongelmaksi tulee se, että jos projekti on iso ja sovelluksessa on useita taustasäikeitä tekemässä operaatioita samaan aikaan, on niiden hallitseminen vaikeata. Coroutines-ominaisuus mahdollistaa vaativien ja pitkäaikaisten operaatioiden suorituksen ilman, että pääsäikeen toiminta estyy. Ne ovat myös kevyitä ja vähän muistia kuluttavia verrattuna tavallisiin säikeisiin. Coroutines-ominaisuus on jatkuvassa kehityksessä, joten sen paketti on merkittynä kokeelliseksi ja niiden käyttämiseksi ne on haettava kotlin.coroutines.experimental-paketista. [30.]

Kotlinia pidetään objektiorientoituneena kielenä, vaikka siihen on sisällytetty funktionaalisen ohjelmoinnin ominaisuuksiakin, esim. korkean järjestyksen funktiot eli lambdat. Projekti, joka käyttää Kotlinia tiedostoissaan ja kirjastoissaan, pystyy pitämään sisällään myös Java-koodia, sillä Kotlin on yhteensopiva Javan kanssa. Kotlinin muuttuessa Androidin viralliseksi ohjelmointikieleksi on Android Studio-ohjelmointiympäristöön myös lisätty Kotlin-tuki. Tuki sisältää jopa vaihtoehdon muuttaa Java-tiedosto Kotlin-tiedostoksi yhdellä painikkeen painalluksella. [14.]

Koska Kotlin on turvallisempaa ja sitä voidaan käyttää Javan kanssa samassa sovelluksessa, se on suositeltu ohjelmointikieli sovelluskehitykseen. Uusille kehittäjille Kotlin voi olla välillä haastavaa, sillä siitä ei löydy vielä paljoa esimerkkejä, niin kuin Javasta. Android on jo päivittänyt omat dokumenttinsa myös Kotlin-esimerkeillä. Vuoden 2018 alussa käynnissä olevan Kotlinin tilaa käsittelevän tutkimuksen alkutuloksien mukaan 40 % kyselyyn vastanneista aloitti Kotlinin opettelemisen sen jälkeen, kun Google teki siitä virallisen Androidin ohjelmointikielen. [31.]

Kotlinin kehittämisen ohella JetBrains on myös julkaissut uuden projektin nimeltä Kotlin Native. Se mahdollistaa Kotlinin käytön eri alustoilla, kuten iOS:ssa ja web-kehityksessä. Koska Kotlin Native on esi-alfa vaiheessa, sitä ei vielä kannata ottaa käyttöön oikeissa projekteissa. Kotlinin kehittäjän Andrey Breslavin mukaan tavoitteena on parin vuoden sisällä korvata JavaScript-ohjelmointikieli kokonaan Kotlin Nativella web-kehityksessä. [25.]

4 Sovelluksen suunnittelu

Insinöörityön tarkoitus oli esitellä Kotlin-ohjelmointikielen vahvuuksia sovelluskehityksessä Androidille ja vertailla hieman, miten se eroaa Java-ohjelmointikielestä. Työn osana tehtiin sovellus helpottamaan Reddit-uutispalvelun lukemista Android-puhelimilla. Reddit on vuonna 2005 perustettu sosiaalisen median verkkosivusto, jossa käyttäjät jakavat linkkejä, kuvia, artikkeleita ja videoita. Reddit on paikka, jossa on paljon keskustelua ajankohtaisista asioista ja niistä asioista, jotka käyttäjiä kiinnostavat. Keskustelukanavia on hyvin paljon, ja ne kohdentavat julkaisut tiettyyn asiaan, esimerkiksi politiikkaan, ruuanlaittoon tai vaikka tiettyyn tietokonepeliin. Käyttäjät äänestävät julkaisuja ylöspäin tai alaspäin, ja tämä mahdollistaa suosituimpien aiheiden näkymisen etusivulla. Redditin sisältö ei rajoitu vain julkaisuihin, vaan joka julkaisussa on kommenttiosio, jossa käyttäjät keskustelevat julkaisusta.

Sovelluksen valmistelu

Sovellusta varten, joka käyttää Redditin REST-rajapintaa, on oltava sovelluksen tiedot täytettynä Redditin sivuilla Sovellukset-osiossa. Tiedoissa pitää olla sovelluksen nimi, kuvaus, kuvauksen verkkosivu, sovelluksen tunnus ja uudelleenohjauksen verkkosivu, joita tarvitaan, kun noudetaan käyttöoikeustietueen koodia.

Sovelluksen kuvaukseen tarkoitettu verkkosivu on ”http://example.com” ja uudelleenohjaukseen ”http://www.example.com/my_redirect". Example-verkkotunnukset on tarkoitettu käytettäväksi esimerkkeihin, joissa tarvitaan verkkosivua.

Sovelluksessa käytetään kuvakkeita ja ne on etsittävä ja ladattava Googlen virallisesta Material.io-sivustosta. Sieltä löydettävät kuvakkeet ovat Androidin materiaaliohjesääntöjä noudattavia ja niitä käytetään materiaaliohjesääntöjä noudattavissa Android-sovelluksissa.

Sovellusten suunnitteluvaiheessa on hyvä jakaa tekovaiheet ja sovelluksen ominaisuudet osiin. Tämän jälkeen jaetaan projektin teon aikaväli sopivankokoisiin kappaleisiin. Tämän projektin teossa käytettiin Trello-palvelua, joka mahdollistaa edellä mainitun visualisoimisen. Projektin teko jaettiin noin yhden viikon pituisiin jaksoihin, ja jokaiseen jaksoon sisällytettiin jokin sovelluksen ominaisuus. Näin pystytään hallitsemaan projektin tekoa yksin tai ryhmässä.

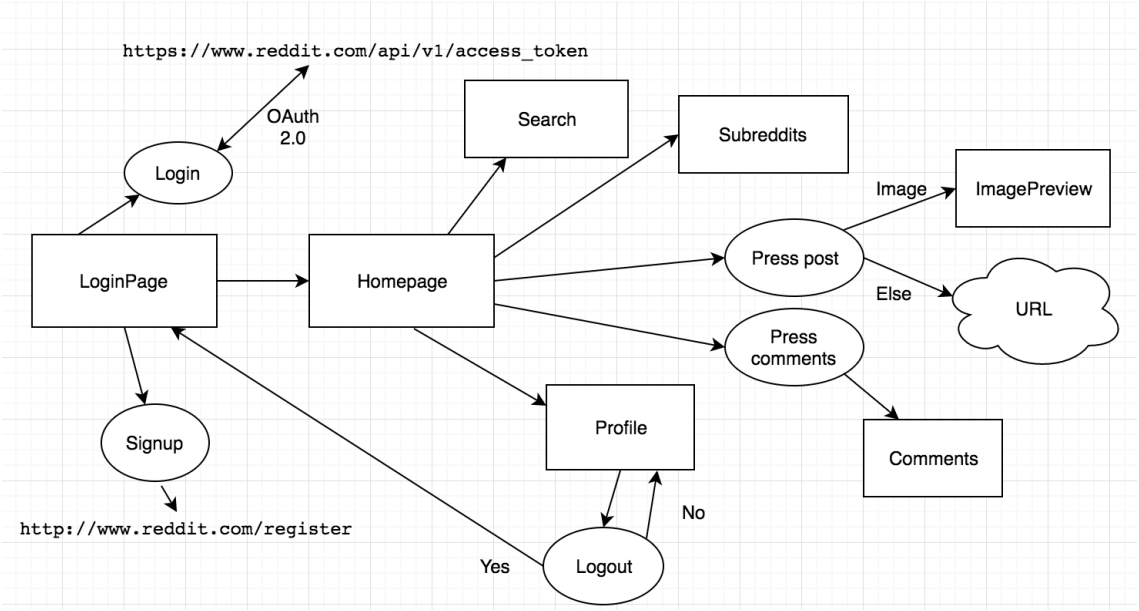
Sovelluksen teossa on myös huolehdittava koodin versionhallinnasta. Versionhallinta on hyvin tärkeää, ja projektissa käytettiin siihen GitHub-versionhallintasivua. Valmiiden uusien ominaisuuksien teon jälkeen on suositeltavaa tallentaa projektin koodi johonkin versionhallintasivulle. Tämä mahdollistaa koodin palauttamisen eri versioihin ja sen pitämisen turvassa pilvipalvelussa.

Sovelluksen ohjelmointirajapinta

Ohjelmointirajapinta on määritelmä, jolla sovellukset keskustelevat toistensa kanssa. Tässä projektissa käytettiin Redditin omaa ohjelmointirajapintaa, josta haettiin julkaisuja, kommentteja ja muita tietoja. Redditin ohjelmointirajapinta on hyvin laaja, ja siksi se on erinomainen sovelluksien teossa. Tämä sovellus ei ole kaupallinen, joten se tarvitsi käyttöön vain muutamia päätepisteitä. Jotkut näistä päätepisteistä tarvitsevat käyttöoikeustietueen sisällytettynä kutsun ylätunnisteeseen, sillä niissä haetaan henkilökohtaisia tietoja. Sovelluksessa tarvittiin päätepisteitä keskustelukanavien julkaisujen hakemiseen, käyttäjän tilattujen keskustelukanavien hakemiseen, käyttäjän tietojen hakemiseen ja julkaisun kommenttien hakemiseen.

Sovelluksen ominaisuudet

Sovelluksen ominaisuudet ja niiden kulku ovat nähtävissä kuvassa 6. Ensiksi sovelluksessa täytyy kirjautua Reddit-palveluun ja antaa sovellukselle lupa käyttää omia Reddit-käyttäjän tietoja. Tämän jälkeen suoritetaan OAuth 2.0 -autorisoinnilla käyttäjälle käyttöoikeustietue rajapinnan kanssa kommunikoimiseen. Kirjautumisen jälkeen käyttäjä on valmis käyttämään sovellusta. Sovelluksen pohjana toimii kolme eri Androidin fragmenttia, joiden nimet ovat Homepage, Subreddits ja Profile. Fragmenttien vaihtamiseen käytetään Androidin alapalkkinavigaatiota. Sovelluksen yläpalkissa on hakuominaisuus, jolla käyttäjä pystyy etsimään keskustelukanavia, ja onnistuneen haun jälkeen sovellus siirtyy Homepage-näkymään, ja siellä pääsee lukemaan haetun keskustelukanavan sisältöä.



Kuva 6. Insinöörityön sovelluksen kulkukaavio.

Homepage-näkymä pitää sisällään 25 suosituinta julkaisua Reddit-palvelussa. Julkaisua painettaessa se vie julkaisun web-sivulle, mutta jos julkaisun verkkotunnus on ”i.redd.it”, käyttäjä ohjataan sovelluksen kuvien näyttämiseen tehtyyn aktiviteettiin. Aktiviteetissa julkaisun kuvan web-osoitteesta saatu kuvatiedosto piirretään käyttäjälle nähtäväksi Picasso-kirjastolla. Kommentti-kuvaketta painamalla käyttäjä pääsee lukemaan julkaisun 25 suosituinta kommenttia kommenttien lukemiseen tarkoitetussa aktiviteetissa.

Subreddits-näkymässä näytetään käyttäjän tilatut keskustelukanavat, ja niitä painamalla sovellus ohjautuu Homepage-näkymään, ja siellä on 25 keskustelukanavan suosituinta julkaisua. Profile-näkymässä käyttäjä pystyy tarkastelemaan omia tietojaan. Tiedoissa on käyttäjän oma käyttäjänimi, kommenteista saatu karma, julkaisuista saatu karma, Reddit-kulta, käyttäjän tekopäivämäärä ja tieto siitä, onko käyttäjä verifioitu käyttäjä. Karma on merkki siitä, kuinka paljon käyttäjä on tehnyt Reddit-yhteisön hyväksi kommentoimalla ja julkaisemalla julkaisuja [27]. Reddit-kulta taas on tapa tukea Reddit-palvelua, ja sillä saa erikoisominaisuuksia, esimerkiksi kommenttien korostaminen ja erikoiskäyttäjäkuvat. Profile-näkymässä on myös uloskirjautumiseen tarkoitettu painike, josta painamalla käyttäjä kirjataan ulos ja käyttöoikeustietue pyyhitään pois puhelimen muistista.

Sovelluksessa käytetyt tärkeimmät kirjastot

Sovelluksissa käytetään usein kolmannen osapuolen kirjastoja, joilla saadaan käyttöön ohjelmointia helpottavia ominaisuuksia ja muiden innovatiivisia ideoita. Kirjastojen käyttämisessä on hyvä muistaa, että joitakin kirjastoja ei enää päivitetä ja niissä voi tulla tämän myötä yhteensopivuus ongelmia uusimpien Android-versioiden kanssa. Projektissa tutkittiin suosituimpia ja hyödyllisimpiä kirjastoja ja päädyttiin seuraaviin kirjastoihin:

Retrofit on HTTP-asiakasohjelma, joka muuttaa HTTP-pyynnöt Java-rajapinnoiksi. Kotlin mahdollistaa Java-yhteensopivuuden takia Java-kirjastojen toimivuuden. Rajapinnan lähettämien JSON-tiedostojen tallentamiseen objekteihin Retrofit tarvitsee GSON-konvertterin, joka tässä projektissa oli Retrofit2-converter-gson. Sovellus käyttää Retrofit-kirjastoa kaikkien API-kutsujen käsittelyssä.

OkHttp on myös HTTP-asiakasohjelma, jota Retrofit käyttää kutsuissaan, jos mahdollista. Tässä projektissa sitä käytettiin autentikointiin.

RxJava on Javan virtuaalikoneelle tehty versio Reactive Extensions -kirjastolle. Tämä kirjasto on käyttöliittymä epäsynkronoidulle ohjelmoinnille, ja se käyttää tarkkailija-suunnittelumallin parhaita puolia yhdistettynä funktionaaliseen ohjelmointiin. RxJava-kirjastoa käytettiin tässä projektissa API-kutsujen vastauksien käsittelyssä.

Anko Commons on yksi osa Anko-kirjastosta. Anko on Kotlinille tehty kirjasto, joka tarjoaa paljon ohjelmointia helpottavia työkaluja, ja tekee koodista puhtaampaa. Anko Commons on tarkoitettu mm. aktiviteetin avaamiselle, dialogeille ja toast-toiminnoille. Sovelluksessa sitä käytettiin edellä mainittuihin asioihin, mikä mahdollisti siistimmän koodin.

Picasso on kirjasto, joka on tarkoitettu kuvien lataamiseen internetistä ja niiden näyttämiseen sovelluksessa. Picasso hallitsee automaattisen muistin- ja levynhallinnan. Projektissa sitä käytettiin julkaisujen kuvien ja näytekuvien näyttämiseen.

PrettyTime on kirjasto, jolla saa formatoitua päivämääräobjektit helposti niin, että tiedetään, montako tuntia tai minuuttia on kulunut. Tällä näytetään projektissa julkaisujen luontipäivämäärä.

5 Sovelluksen toteutus

Kehitysympäristö

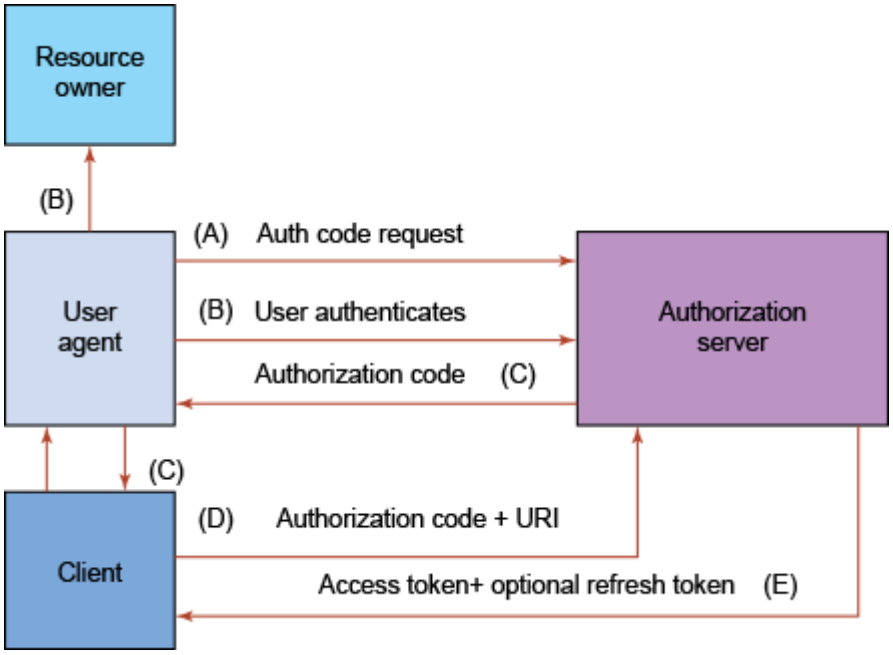
Projektin suunnitteluvaiheen jälkeen alkoi sovelluksen tekeminen. Ohjelmointiympäristönä toimi Android Studio -ohjelma. Se on Androidia varten tehty virallinen ohjelmointiympäristö. Sen avulla pystyy vaivattomasti kehittämään sovelluksia Androidin virallisilla ohjelmointikielillä ja Androidin omaa SDK:ta käyttäen. Lisäksi Android Studiossa on emulaattoriominaisuus, jolla pystyy testaamaan sovellusta ilman fyysistä mobiililaitetta. Emulaattori on kuin oikea mobiililaite näkyvällä käyttöliittymällä, mutta se toimii tietokoneessa. Android Studio tarjoaa paljon erimallisia mobiililaitteita, ja käyttäjä pystyy myös itse valitsemaan, mikä Android-versio niissä on. Sovellusta testattiin kehittäjän omassa OnePlus 3 -puhelimessa, toisessa testipuhelimessa Samsung Galaxyssa ja Android Studion emulaattorilla. Projektin sovelluksessa käytetty minimi-SDK-versio oli 23 ja kohde SDK-versio 26. Sovelluksen rakennustyökaluina käytettiin Gradlen versiota 3.0.1 ja Kotlinin versiota 1.1.60.

Kirjautuminen OAuth 2.0 -autentikaatiolla

OAuth 2.0 on toinen versio OAuth-autorisointiprotokollasta. Se on yksinkertaistettu sen alkuperäisestä versiosta, ja sitä käytetään käyttäjän tunnistautumisessa. OAuth 2.0:n suosiosta kertoo se, että se on käytössä monessa suositussa sosiaalisen median sovelluksessa [22]. Autentikoinnilla tarkoitetaan käyttäjän todennusta, eli sillä selvitetään, onko käyttäjä valtuutettu sovelluksen käyttäjä [23]. Autorisoinnilla tarkoitetaan käyttäjän valtuuksien tarkistamista, eli onko käyttäjällä oikeuksia suorittaa tiettyjä kutsuja ja toimenpiteitä [23].

OAuth 2.0 -protokollalla on neljä erilaista roolia ja valtuutustapaa, joita sillä voidaan käyttää. Rooleja ovat resurssinhaltija, resurssipalvelin, asiakassovellus ja valtuutuspalvelin. Projektissa resurssipalvelin ja valtuutuspalvelin olivat Redditin ja niiden käyttöä varten asiakassovellus oli projektin sovellus. Resurssin haltija oli sekä Reddit että sovelluksen käyttäjä [23].

Valtuutustapana Reddit API käyttää Authorization Code Grant -nimistä menetelmää, jota käytetään eniten eri valtuutustavoista (kuva 7).



Kuva 7. OAuth 2.0 Authorization Code Grant-valtuutustapa [24].

Reddit API:n kanssa kommunikointiin sovelluksen käyttäjä tarvitsee Reddit-käyttäjätilin. Sovellus mahdollistaa käyttäjätilin rekisteröimisen painamalla sovelluksen ensimmäisessä näkymässä Signup-painiketta. Reddit API:n käyttöön sovellus tarvitsee käyttöoikeustietueen eli merkkijonon, jota käytetään tiettyjen API-kutsujen kanssa, jotta Reddit API tietää, kenen käyttäjän ja minkä sovelluksen kanssa se kommunikoi.

Sovelluksessa OAuth 2.0 -autentikointi tehdään LoginActivity-luokassa. Se on sovelluksen aloituspiste, kun sen avaa mobiililaitteessa. Login-painiketta painettaessa sovellus ohjaa käyttäjän Redditin omaan autentikointiosoitteeseen (esimerkkikoodi 6), joka sisältää web-osoitteen ”https://www.reddit.com/api/v1/authorize.compact?”, ja tätä osoitetta jatketaan asiakastunnuksella, vastauksen tyypillä, joka on tässä tapauksessa koodi, sillä sovellus haluaa vastauksena autorisointikoodin. Osoitteessa on myös tila, joka on ”LOGIN”, koska halutaan kirjautua Redditin palveluun. Lopuksi osoitteeseen laitetaan uudelleenohjausosoite, joka laitetaan Redditin-sivuilta oman sovelluksen asetuksista, koodin kesto, joka on jatkuvaa tyyppiä, ja viimeiseksi sovelluksen alat eli mitä aloja sovellus käyttää. Aloja ovat esimerkiksi henkilöllisyys, lukeminen ja omat suosikkikeskustelukanavat. Redditin API-dokumentaatiosta löytyvät tiedot, mitä aloja API-kutsut käyttävät.

```
https://www.reddit.com/api/v1/authorize.compact?client_id=$CLIENT_ID&res-
ponse_type=code&state=$STATE&redirect_uri=$REDIRECT_URI&duration=per-
manent&scope=$SCOPE
```

Esimerkkikoodi 6. Lopullinen autorisointiosoite.

Käyttäjä palaa sovellukseen, ja luokan onResume-metodissa vastaanotettu koodi tallennetaan ja syötetään getAccessToken-metodiin, jolla haetaan sovelluksen lopullinen käyttöoikeustietue ja sen päivittämiseen tarkoitettu päivitystietue. Metodissa luodaan OkHttpClient-objekti ja uusi pyyntö-objekti, jonka avulla sovellus tekee pyynnön noutaa tietueet. Pyyntö-objekti pitää sisällään räätälöidyn ylätunnuksen ja käyttöoikeustietueen hakuun tarkoitetun osoitteen, johon on sisällytetty edellä mainittu haettu koodi. Onnistuneen käyttöoikeustietueen ja päivitystietueen saamisen jälkeen sovellus siirtyy MainActivity-luokkaan ja tallentaa tietueet sekä avaa uuden käyttäjäsession.

Kommunikointi REST-rajapinnan kanssa

Rajapinnan ja sovelluksen välille on saatava yhteys, jotta ne pystyvät kommunikoimaan keskenään. Tätä varten sovelluksessa käytettiin Retrofit-kirjastoa. Kun sovelluksessa ollaan tekemisissä verkon käytön kanssa, on Android Manifest -tiedostoon laitettava lupa sovelluksen verkon käytölle. Reddit-rajapinnan dokumentaatio tarjoaa valtavan määrän eri tarkoituksiin tehtyjä päätepisteitä ja niiden käytön ohjeet. Tämän projektin sovellus tarvitsi vain muutamaa niistä.

API-kutsujen hallintaan sovellukseen tehtiin APIManager-luokka. APIManager-luokan sisällä oleva create-funktio luo Retrofit-objektin, jossa määritellään sen käyttämä web-osoite API-kutsuille, ja siellä myös lisätään tarvittavat JSON:n konvertoimiseen ja kutsujen tekoon tarvittavat adapterit. Tämä objekti annetaan sovelluksessa kaikkien luokkien esittäjä-luokille, jotka tekevät API-kutsuja. Jotkin API-kutsut, esim. käyttäjän keskustelukanavien ja profiilitietojen hakemiseen tarkoitetut, vaativat käyttäjän käyttöoikeustietueen, joka on haettu kirjautumisen yhteydessä ja tallennettu muistiin. Näitä kutsuja varten tehtiin toinen luokka, APIManagerWithInterceptors, joka on muuten identtinen, paitsi että siellä on web-osoite, joka vaaditaan kyseisten kutsujen tekoon ja räätälöity Interceptor-luokka nimeltä OAuthInterceptor (esimerkkikoodi 7).

```
class OAuthInterceptor: Interceptor {
     val token = UserSession.accessToken

     override fun intercept(chain: Interceptor.Chain): Response {
          val request = chain.request().newBuilder()
                    .addHeader(”User-Agent”, ”ReadIt”)
```

```
                    .addHeader(”Authorization”, ”bearer ” + token)
                    .build()
            return chain.proceed(request)
        }
}
```

Esimerkkikoodi 7. Sovelluksen OAuthInterceptor-luokka.

Interceptor-luokka on OkHttp-kirjaston ja sillä lisätään API-kutsuihin räätälöity ylätunniste. Räätälöity ylätunniste sisältää autorisointitunnisteen, jonka parametrinä on käyttäjän käyttöoikeustietue. Tämän avulla Redditin rajapinta tietää, minkä sovelluksen kanssa se kommunikoi, mikä mahdollistaa käyttäjän henkilökohtaisten tietojen hakemisen.

Sovelluksen API-rajapintaluokka sisältää Retrofitiä varten päätepisteet ja niiden kutsuun tarkoitetut funktiot.

Esimerkkikoodissa 8 on nähtävissä, että funktio on tarkoitettu keskustelukanavan julkaisujen hakemiseen. Funktio ottaa parametriksi keskustelukanavan nimen ja tämän jälkeen yhdistää sen kutsun polkuun, joka on merkitty Retrofitin toimesta sivuhuomautuksena. GET-etuliitettä käytetään, kun haetaan dataa REST-rajapinnoista. Tämä funktio palauttaa Observable-objektin, joka sisältää halutun datamallin. Observable-objekti tulee RXJava-kirjaston mukana. RXJava käyttää hyväksi tarkkailija-mallia, jossa on tarkkailija ja tarkkailtava objekti. Niiden välinen tiedonsiirto tapahtuu tarkkailijaa käyttäen. Tarkkailija-objekti tekee tilauksen tarkkailtavalle objektille. Sen jälkeen tarkkailija reagoi mihin vain, mitä tarkkailtava lähettää.

```
@GET("{subReddit}.json")
fun getSubredditsPosts(@Path("subReddit") subReddit: String): Observable<Sub-
RedditResponse>
```

Esimerkkikoodi 8. Keskustelukanavan julkaisujen hakemiseen tarkoitettu funktio.

RxJava on oivallinen kirjasto epäsynkronoitujen funktioiden hallinnassa. Esimerkkikoodissa 9 nähdään esimerkkinä, miten insinöörityön sovelluksessa haetaan julkaisuja esittäjä-luokassa. Kun käyttäjä avaa sovelluksen julkaisujen näyttämiseen tarkoitetun näkymän, se kutsuu aktiviteetin ja esittäjän teon jälkeen esimerkkikoodin funktiota. Funktio ottaa parametrinä keskustelukanavan nimen ja antaa sen API-objektille, joka tekee sillä kutsun Redditin rajapinnalle. Kutsun jälkeen se käyttää hyväksi RxJavan Subscribe-ominaisuutta, jolla pystyy hallitsemaan ja ohjaamaan dataa. Kutsun vastaus on tilattu

Schedulers.io-objektia käyttäen, jota käytetään epäsynkronoidussa suorituksessa. Tarkkailu tapahtuu Androidin pääsäikeessä, ja se on kiinnitetty palautettuun Observable-objektiin.

Kun funktio on tehnyt API-kutsun, se odottaa Observable-objektia, joka sisältää kyseisen keskustelukanavan 25 julkaisun datan. Result-vaihtoehto tapahtuu, kun vastaus on onnistunut. Silloin onnistuneen vastauksen data puretaan Kotlinin map-funktiota käyttäen. Tämän jälkeen se annetaan parametrina näkymän showPosts-funktiolle. Koska näkymä on passiivinen näkymä, se vain siirtää datan aktiviteettin tai fragmentiin, jossa se näytetään käyttäjälle. Error-vaihtoehto on sitä varten, että kutsu epäonnistuu ja vastauksena tulee virheilmoitus. Se näytetään halutusti joko käyttäjälle tai kehittäjän lokitiedostoissa.

```
fun fetchRedditPosts(subReddit: String) {
      api.getSubredditsPosts(subReddit)
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe(
                  {result ->
                        val allData = result.allData
                        val allChildrens = allData.children.map { it }
                        val allChildrensData = allChildrens.map { it.data }
                        view.showPosts(allChildrensData)
            },
            { error ->
                        view.showError(error.toString())
            }
      )
}
```

Esimerkkikoodi 9. Esittäjän funktio hakea julkaisuja ja käsitellä niiden vastaukset.

Kotlinin ominaisuuksien käyttö sovelluksessa

Sovelluksessa käytettiin Kotlinin laajennos-ominaisuutta edistymispalkin näyttämiseen ja piilottamiseen sekä päivämäärän muokkaamiseen. Käyttökokemuksen parantamista varten sovelluksessa on edistymispalkki näyttämässä, milloin sovellus lataa sisältöä, ja se poistuu, kun sisältö on ladattu ja näytetty käyttäjälle. Laajennos tehtiin Androidin näkymä-luokalle, johon tehtiin laajennos-funktio sen näyttämiseen ja piilottamiseen. Nyt sovelluksessa pystytään piilottamaan ja näyttämään mikä vain näkymä käyttämällä tätä laajennosta. Esimerkkikoodissa 10 näkyy Date-luokalle tehty laajennos, joka muuttaa Date-objektin String-muotoon käyttäen vuosiformaattia. Tämän jälkeen muokattu String-muodossa oleva päivämäärä palautetaan. Tätä laajennosfunktiota käytetään Profile-näkymässä, kun näytetään käyttäjän rekisteröintipäivämäärä.

```
fun Date.format(context: Context): String {
val flags = DateUtils.FORMAT_SHOW_YEAR
return DateUtils.formatDateTime(context, time, flags)
}
```

Esimerkkikoodi 10. Date-luokan laajennoksen format-funktio.

Kotlinin Kotlin Android Extension-ominaisuus mahdollistaa helpomman tavan saada referenssi näkymiin. Android-kehittäjille on tuttua hakea referenssi näkymään findViewById-metodilla, ja tätä kehittäjän piti kutsua, joka kerta, kun haluttiin olla tekemisissä kyseisen näkymän kanssa. Kotlin Android Extension-ominaisuudella ei tarvitse luoda referenssiä, vaan ne on luotu jo automaattisesti. Kehittäjä vain kutsuu kyseisen näkymän id:tä ja voi suoraan jo kutsua sen funktioita tai tehdä sille muutoksia [28]. Esimerkkikoodissa 11 on esimerkki Kotlin Android Extension-ominaisuuden käytöstä projektin sovelluksessa.

```
rv_home.adapter = adapter
```

Esimerkkikoodi 11. Homepage-näkymän adapterin laittaminen RecyclerView-komponenttiin.

Esimerkkikoodissa 12 on nähtävissä, miten adapteri laitetaan RecyclerView-komponenttiin Javassa. Kotlin ei tarvitse muuttujaa komponentille eikä etsi näkymää findViewById-funktiolla. Myös setAdapter-funktio on lyhennettynä Kotlinissa pelkäksi adapteriksi.

```
private RecyclerView mRecyclerView;
mRecyclerView = (RecyclerView)findViewById(R.id.rv_home);
mRecyclerView.setAdapter(mAdapter);
```

Esimerkkikoodi 12. Adapterin laittaminen RecyclerView-komponenttiin Javalla.

Vaikka Kotlinin menetelmä säästää vain pari riviä koodia, niin se on silti hyvä esimerkki siitä, miten asioita yksinkertaistetaan ja kehittäjien työtä tehdään helpommaksi. Kotlin Android Extensions tallentaa myös näkymät välimuistiin, ja ensimmäisen näkymän käyttökerran jälkeen se hakee sen suoraan välimuistista. Tämä nopeuttaa sovelluksen toimivuutta. [29.]

Sovellus saa dataa hakiessaan vastauksen Redditin rajapinnasta JSON-muodossa, joka on hyvin suosittu tiedostotyyppi palauttaa dataa. Sovelluksen APIManagerissa on sille annettu adapteriksi GSON-adapteri, joka mahdollistaa JSON:n sijoittamisen Java-objekteihin. Esimerkkikoodista 13 näkyy miten sovelluksessa käytetään GSON-huomiointia.

```
class SubRedditResponse {
      @SerializedName(“data”)

      val allData: AllData = AllData()

}
```

Esimerkkikoodi 13. Sovelluksen SubRedditResponse-luokka.

API-kutsusta saadun JSON-vastauksen sisällä on haluttua dataa, joka löytyy data-objektin sisältä. Sovelluksessa tallennetaan datan sisältö AllData-luokkaan. ”@SerializedName”-huomioinnilla pystytään sijoittamaan halutun JSON:n sisällä oleva data itse määriteltyyn muuttujaan.

6 Jatkokehitysmahdollisuudet ja yhteenveto

Insinöörityönä tehdyn sovelluksen tulevaisuuden jatkokehitysmahdollisuudet ovat laajat:

Käyttöoikeustietueen päivittäminen. Nykyisellään sovellus hakee yhden käyttöoikeustietueen sovelluksen käyttöä varten ja se menee vanhaksi yhdessä tunnissa. Tämän jälkeen käyttäjäsessioon tallennettua päivitystietuetta käyttäen pitäisi hakea uusi käyttöoikeustietue Redditin rajapinnasta, minkä jälkeen sitä voi taas käyttää tunnin.

Julkaisun kommentit mahdollistavat erilaisten toiminnallisuuksien implementoinnin. Sovellus näyttää tässä versiossa 25 suosituinta kommenttia, mutta jatkokehityksessä voisi lisätä mahdollisuuden ladata lisää kommentteja ja vaihtoehdon vastata kommentteihin sekä kirjoittaa uusia kommentteja.

Yleinen toiminnallisuuksien lisääminen käyttäen hyväksi Redditin laajaa API-rajapintaa. Sovellukseen pystyy lisäämään esimerkiksi julkaisujen äänestämisfunktiot, julkaisujen jakamisen sosiaalisessa mediassa tai muiden käyttäjien tutkimisen.

Sovelluksen teko muille käyttöjärjestelmille. Sovellus on nyt saatavilla vain Android-laitteille. Jatkokehitysmahdollisuutena on sovelluksen tekeminen myös iOS-laitteille: joko kokonaan oma sovellus Swift- tai Objective-C-kieltä käyttäen tai Kotlin Nativea hyödyntäen sovellus myös iOS:lle.

Yhteenveto

Insinöörityössä perehdyttiin Kotlin-ohjelmointikieleen ja sen käyttämiseen Android-sovelluksen teossa sekä Reddit-uutispalveluun ja sen rajapinnan käyttämiseen. Työssä rakennettiin myös uutissovellus Reddit-palveluun. Sovelluksen avulla Reddit-palveluun rekisteröitynyt käyttäjä pääsee tutkimaan palvelua ja lukemaan mielenkiintoisia ja ajankohtaisia julkaisuja. Sovellus tehtiin MVP-arkkitehtuurimallilla, joka mahdollistaa sovelluksen jatkokehityksen sulavasti. Muiden kehittäjien on helppo kehittää sovellukseen uusia ominaisuuksia ja lukea koodia. Jatkokehityksen aikana suoritettaisiin käyttäjätestaamista vapaaehtoisilla. Tämä mahdollistaisi uusien ajatusten syntymisen ja käyttäjäkokemuksen parantamisen. Jatkokehityksen jälkeen sovellus voidaan julkaista Google Play Storessa, joka on Android-sovellusten virallinen kauppapaikka.

Minulle uusina asioina tulivat Redditin rajapinnan kanssa työskenteleminen ja OAuth 2.0 -autorisoinnin käyttäminen sovelluksessa. Kotlinin ominaisuuksien ja sen avulla turvallisen koodin hyödyntäminen mahdollisti sujuvan kehittämisen ja useilta virheilmoituksilta välttymisen. Lopputuloksena sovelluksesta tuli toimiva, nopea ja skaalautuva. Kaikki halutut toiminallisuudet saatiin tehtyä, ja Kotlinin käyttäminen projektissa onnistui halutulla tavalla.

Lähteet

1. Callaham, John. 2017. The history of Android OS: its name, origin and name. Verkkoaineisto. Android Authority. https://www.androidauthority.com/history-android-os-name-789433/. Luettu 7.2.2018.

2. Dobie, Alex; Holly, Russel & Hildenbrand, Jerry. Android Early Days. Verkkoaineisto. Android Central. https://www.androidcentral.com/androids-early-days. Luettu 7.2.2018.

3. Android overtakes Windows for first time – Statcountter. 2017. Verkkoaineisto. Business Wire. https://www.business-wire.com/news/home/20170403005635/en/Android-Overtakes-Windows-Time-%E2%80%93-StatCounter. Luettu 9.2.2018.

4. Sharma, Neha. 2013. The beginner's guide to Android: Android architecture. Verkkoaineisto. Edureka. https://www.edureka.co/blog/beginners-guide-android-architecture/. Luettu 9.2.2018.

5. Hildenbrand, Jerry. 2012. What is kernel? Verkkoaineisto. Androidcentral. https://www.androidcentral.com/android-z-what-kernel. Luettu 9.2.2018.

6. Jain, Sumit. 2014. Android System Architecture. Verkkoaineisto. Android.tutorialhorizon. http://android.tutorialhorizon.com/android-system-architecture/. Luettu 9.2.2018.

7. What do you know about Android Runtime. 2014. Verkkoaineisto. Edc4it. http://www.edc4it.com/blog/mobile/what-do-you-know-about-android-runtime.html. Luettu 9.2.2018.

8. Schenck, Stephen. 2014. Android codebase makes the shift from Dalvik to ART. Verkkoaineisto. Pocketnow. http://pocketnow.com/2014/06/19/art-replaces-dalvik. Luettu 9.2.2018.

9. Burton, Michael. 2015. Android App Development for Dummies. 3rd edition. Wiley.

10. Application fundamentals. Verkkoaineisto. Android. https://developer.android.com/guide/components/fundamentals.html. Luettu 14.2.2018.

11. Android Activity Lifecycle. Verkkoaineisto. Javapoint. https://www.javatpoint.com/android-life-cycle-of-activity. Luettu 14.2.2018.

12. Soni, Nikhil. 2017. Android Activity Explained. Verkkoaineisto. Medium. https://medium.com/@n.nikhil.ns65/android-activity-explained-2b651a8d2c79. Luettu 14.2.2018.

13. Leiva, Antonio. 2016. Kotlin for Android Developers. E-Kirja. Leanpub.

14. Chowdhury, Rahul. 2017. The Ugly Truth about Extension Functions in Kotlin. Verkkoaineisto. AndroidPub. https://android.jlelse.eu/the-ugly-truth-about-extension-functions-in-kotlin-486ec49824f4. Luettu 15.2.2018.

15. Idris, Nazmul. 2018. Deep dive into Android Services. Verkkoaineisto. ProAndroidDev. https://proandroiddev.com/deep-dive-into-android-services-4830b8c9a09. Luettu 26.2.2018.

16. Services. Verkkoaineisto. Android. https://developer.android.com/guide/components/services.html#StartingAService. Luettu 26.2.2018.

17. Learning the Model-View-Controller design pattern in iOS. Verkkoaineisto. LinkedIn. https://www.linkedin.com/learning/learning-the-model-view-controller-design-pattern-in-ios. Luettu 13.3.2018.

18. Witlock, John. 2017. The theory behind Model View Controller. Verkkoaineisto. Mozilla. https://developer.mozilla.org/en-US/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture. Luettu 13.3.2018.

19. Muntenescu, Florina. 2016. Android architecture patterns Part 1: Model-view-controller. Verkkoaineisto. Medium. https://medium.com/upday-devs/android-architecture-patterns-part-1-model-view-controller-3baecef5f2b6. Luettu 13.3.2018.

20. Muntenescu, Florina. 2016. Android architecture patterns Part 2: Model-view-presenter. Verkkoaineisto. Medium. https://medium.com/upday-devs/android-architecture-patterns-part-2-model-view-presenter-8a6faaae14a5. Luettu 13.3.2018.

21. Goutay, Oliver. 2017. Using Espresso and Mockito to test an MVP app. Verkkoaineisto. AndroidPub. https://android.jlelse.eu/using-espresso-and-mockito-to-test-a-mvp-app-d17f751fe966. Luettu 13.3.2018.

22. Bandara, Sathya. 2017. An Introduction to OAuth 2.0. Verkkoaineisto. Medium. https://medium.com/@technospace/an-introduction-to-oauth-2-0-4c71b5fb19ff. Luettu 15.3.2018.

23. TIE-23500 Web-ohjelmointi. Autentikointi. 2015. Verkkoaineisto. Tampereen teknillinen yliopisto. http://www.cs.tut.fi/~seitti/2015/kalvot/auth/all.html. Luettu 15.3.2018.

24. Ojha, Varun. 2014. Authorization code grant. Verkkoaineisto. IBM. https://www.ibm.com/developerworks/library/se-oauthjavapt3/index.html. Luettu 15.3.2018.

25. Breslav, Andrey. 2017. KotlinConf 2017- Deep Dive in to Kotlin/Native by Andrey Breslav. Verkkoaineisto. Jetbrains. https://www.youtube.com/watch?v=3Lqiupxo4CE. Luettu 21.3.2018.

26. Cervone, Francesco. 2017. Model-View-Presenter: Android guidelines. Verkkoaineisto. Medium. https://medium.com/@cervonefrancesco/model-view-presenter-android-guidelines-94970b430ddf. Luettu 23.3.2018.

27. FAQ. Verkkoaineisto. Reddit. https://www.reddit.com/r/help/wiki/faq#wiki_how_does_karma_work.3F. Luettu 26.3.2018.

28. Zhulanow, Yan. Kotlin Android Extensions. Verkkoaineisto. Kotlinlang. https://kotlinlang.org/docs/tutorials/android-plugin.html. Luettu 26.3.2018.

29. Leiva, Antonio. 2017. Kotlin Android Extensions – Say goodbye to findViewById. Verkkoaineisto. AntonioLeiva.com. https://antonioleiva.com/kotlin-android-extensions/. Luettu 28.3.2018.

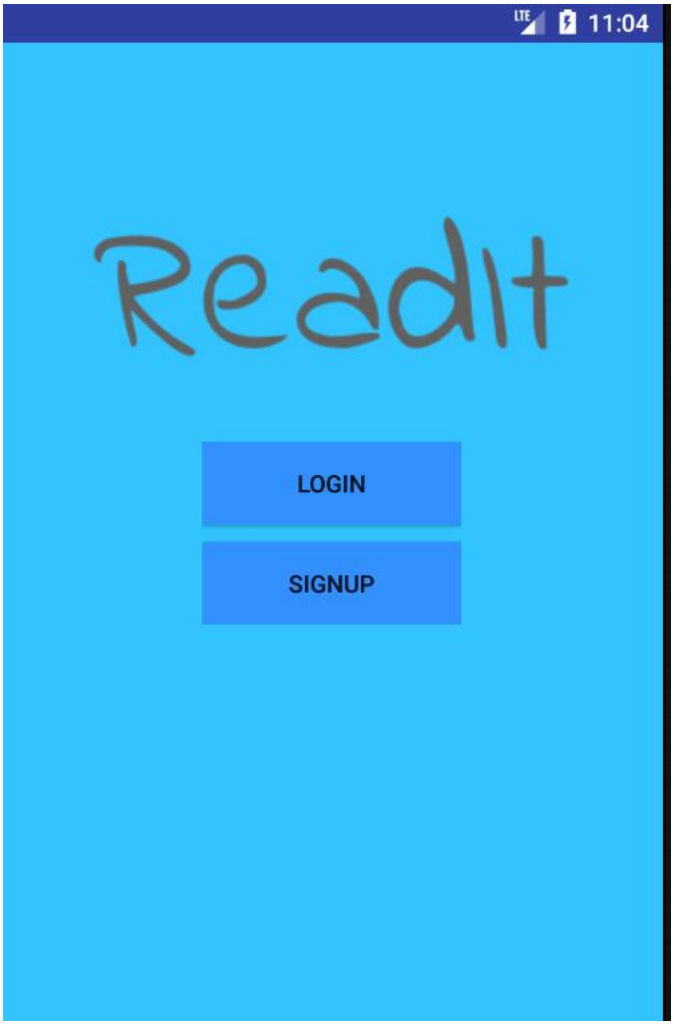
30. Coroutines. Verkkoaineisto. Kotlinlang. https://kotlinlang.org/docs/reference/coroutines.html. Luettu 6.4.2018.

31. Markan, Zan. 2018. Announcing the State of Kotlin Survey. Verkkoaineisto. Pusher. https://blog.pusher.com/announcing-state-kotlin-survey/. Luettu 6.4.2018.

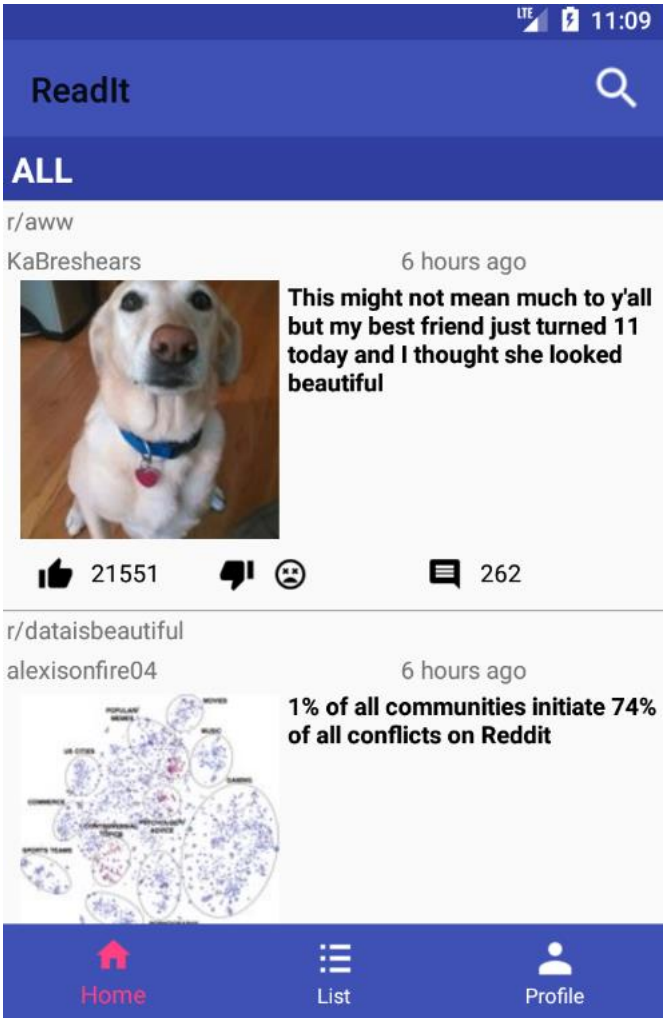
32. Android System Architecture. Verkkoaineisto. Wikimedia. https://commons.wikimedia.org/wiki/File:Android-System-Architecture.svg. Luettu 19.4.2018.

Sovelluksen näkymät (Liitteet 1-5)

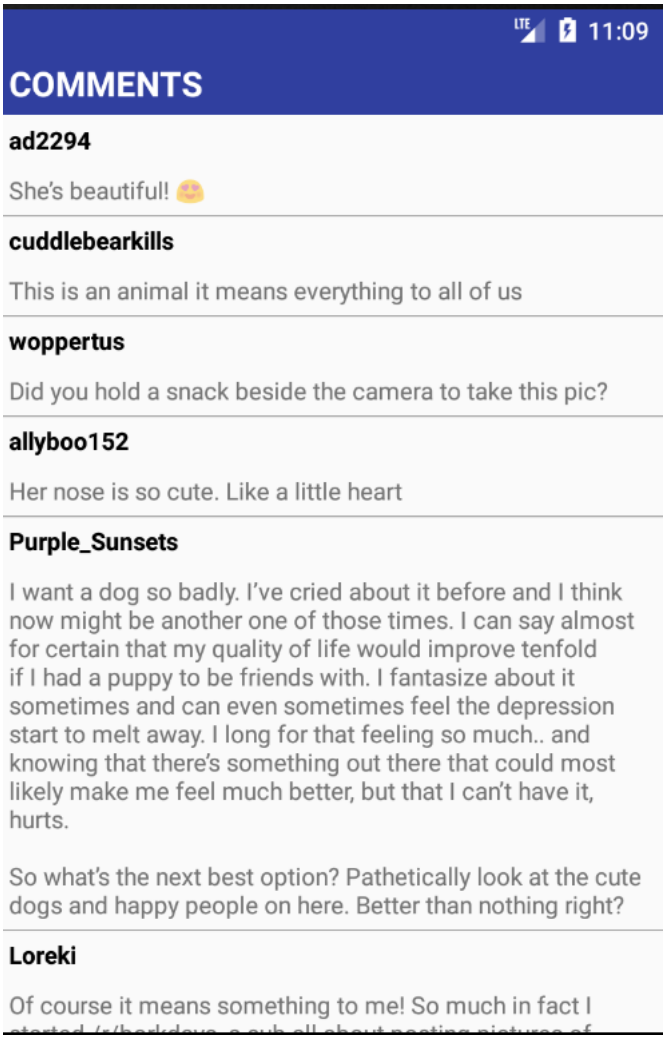
LoginPage-näkymä



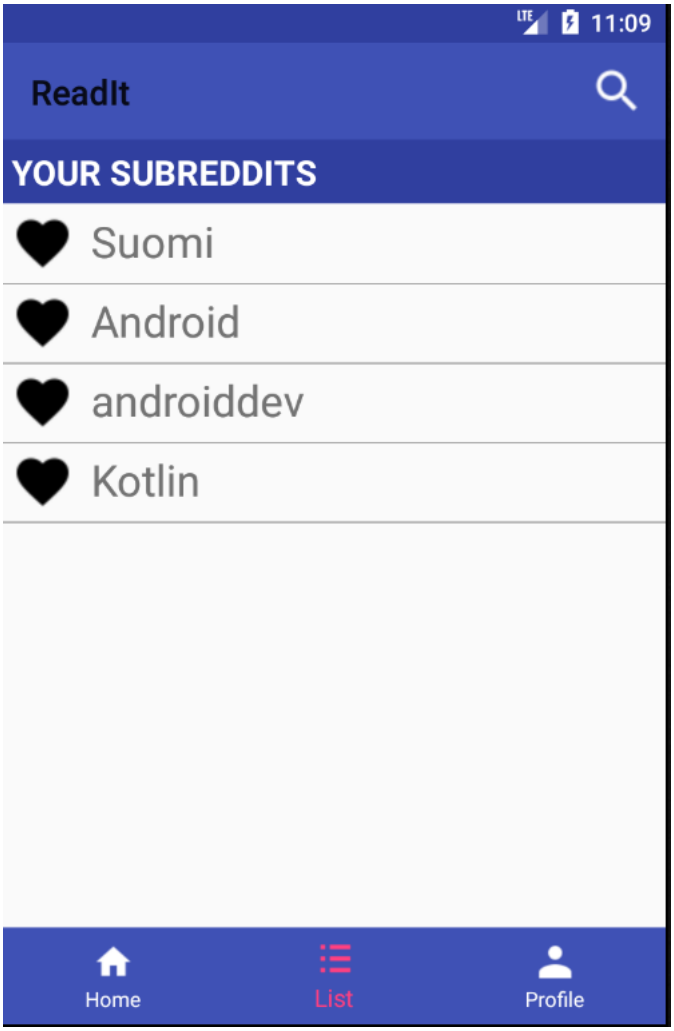
HomePage-näkymä



CommentPage-näkymä



Subreddits-näkymä



ProfilePage-näkymä